Key Transforming Operators in the Mono Class (Part 2)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

• Recognize key Mono operators
  • Concurrency & scheduler operators
• Transforming operators
  • Transform the values and/or types emitted by a Mono
    • e.g., flatMap() & flatMapMany()
Key Transforming Operators in the Mono Class
Key Transforming Operators in the Mono Class

- The `flatMap()` method
- Transform the item emitted by this Mono (asynchronously)

```java
<R> Mono<R> flatMap
    (Function<? super T, ? extends Mono<? extends R>> transformer)
```

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#flatMap](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#flatMap)
Key Transforming Operators in the Mono Class

- The `flatMap()` method
  - Transform the item emitted by this Mono (asynchronously)
  - The param is a function that dynamically binds a new Mono

```java
<R> Mono<R> flatMap
    (Function<? super T, ? extends Mono<? extends R>> transformer)
```

See [docs.oracle.com/javase/8/docs/api/java/util/function/Function.html](docs.oracle.com/javase/8/docs/api/java/util/function/Function.html)
Key Transforming Operators in the Mono Class

- The flatMap() method
- Transform the item emitted by this Mono (asynchronously)
  - The param is a function that dynamically binds a new Mono
- Returns the value emitted by the new Mono

```java
<R> Mono<R> flatMap
    (Function<? super T, ? extends Mono<? extends R>> transformer)
```
Key Transforming Operators in the Mono Class

- The `flatMap()` method
  - Transform the item emitted by this Mono (asynchronously)
  - Can transform the value and/or type of elements it processes
Key Transforming Operators in the Mono Class

- The `flatMap()` method
  - Transform the item emitted by this Mono (asynchronously)
  - Can transform the value and/or type of elements it processes

```java
Mono<BigFraction> bfM1 = makeBigFractionAsync(...);

Mono<BigFraction> bfM2 = bfM1.flatMap(bf ->
    multiplyAsync(bf, sBigReducedFraction))
```

Use `flatMap()` to asynchronously multiply a random `BigFraction` by a large constant & avoid “nested” monos

See Reactive/mono/ex3/src/main/java/MonoEx.java
Key Transforming Operators in the Mono Class

• The flatMap() method
  • Transform the item emitted by this Mono (asynchronously)
  • Can transform the value and/or type of elements it processes

```
Mono<BigFraction> bfM1 = makeBigFractionAsync(...);

Mono<Mono<BigFraction>> bfM2 =
  bfM1.flatMap(bf ->
    multiplyAsync(bf, sBigReducedFraction))
```

This awkward nesting of Monos is needed if flatMap() were replaced by map()!
Key Transforming Operators in the Mono Class

- The `flatMap()` method
  - Transform the item emitted by this Mono (asynchronously)
  - Can transform the value and/or type of elements it processes
- RxJava’s `Single.flatMap()` works the same way

```java
Single<BigFraction> bfS1 = makeBigFractionAsync(...);

Single<BigFraction> bfS2 = bfS1.flatMap(bf ->
    multiplyAsync(bf, sBigReducedFraction))
```

Asynchronously multiply a random `BigFraction` by a large constant

Key Transforming Operators in the Mono Class

- The `flatMap()` method
  - Transform the item emitted by this Mono (asynchronously)
  - Can transform the value and/or type of elements it processes
  - RxJava’s `Single.flatMap()` works the same way
- Similar to the `thenCompose()` method in Java Completable Futures

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#thenCompose](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#thenCompose)
Key Transforming Operators in the Mono Class

- The `flatMapMany()` method
- Transform the item emitted by this Mono into a Publisher

```java
<R> Flux<R> flatMapMany
(Function<? super T, ? extends Mono<? extends R>> transformer)
```

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#flatMapMany](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#flatMapMany)
Key Transforming Operators in the Mono Class

- The flatMapMany() method
- Transform the item emitted by this Mono into a Publisher
- The param is a function that produces a sequence of R

```
<R> Flux<R> flatMapMany
    (Function<? super T,
            ? extends Mono
                <? extends R>>
            transformer)
```

Interface Function<T,R>

Type Parameters:
T - the type of the input to the function
R - the type of the result of the function

All Known Subinterfaces:
UnaryOperator<T>

See docs.oracle.com/javase/8/docs/api/java/util/function/Function.html
Key Transforming Operators in the Mono Class

- The flatMapMany() method
- Transform the item emitted by this Mono into a Publisher
  - The param is a function that produces a sequence of R
  - Returns a Flux that receives the emissions

```java
<R> Flux<R> flatMapMany
    (Function<? super T, ? extends Mono<? extends R>> transformer)
```
15

Key Transforming Operators in the Mono Class

- The flatMapMany() method
  - Transform the item emitted by this Mono into a Publisher
  - Can transform the value and/or type of elements it processes

```
flatMapMany ()
```

![Diagram of flatMapMany() method]
Key Transforming Operators in the Mono Class

- The `flatMapMany()` method
  - Transform the item emitted by this Mono into a Publisher
  - Can transform the value and/or type of elements it processes

```java
Flux<BigFraction> bfF = Flux.fromArray(...);
Mono<BigFraction> bfM = Mono.fromCallable(...);
bfM.flatMapMany(bf1 -> bfF
                .flatMap(bf2 -> Flux
                          .just(bf2)
                          .subscribeOn(Schedulers.parallel())
                          .map(__ -> bf2.multiply(bf1))))...
```

Concurrently transform a Mono result into a Flux result

See Reactive/flux/ex4/src/main/java/FluxEx.java
Key Transforming Operators in the Mono Class

- The flatMapMany() method
  - Transform the item emitted by this Mono into a Publisher
  - Can transform the value and/or type of elements it processes
- RxJava’s flatMapObservable() method in Single is similar

```java
Observable<BigFraction> bfO = Observable.fromArray(...);
Single<BigFraction> bfS = Single.fromCallable(...);
bfS.flatMapObservable(bf1 -> bfO.flatMap(bf2 -> Observable.just(bf2).
    subscribeOn(Schedulers.computation())
    .map(___ -> bf2.multiply(bf1)))).
```

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Single.html#flatMapObservable](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Single.html#flatMapObservable)
End of Key Transforming Operators in the Mono Class (Part 2)